

Gaussian Process Pseudo-Likelihood Models for Sequence Labeling

P. K. Srijith¹, P. Balamurugan², and Shirish Shevade³

¹ Department of Computer Science, University of Sheffield, United Kingdom
pk.srijith@dcs.shef.ac.uk

² SIERRA Project Team, INRIA-ENS, Paris, France
balamurugan.palaniappan@inria.fr

³ Computer Science and Automation, Indian Institute of Science, Bangalore
shirish@csa.iisc.ernet.in

Abstract. Several machine learning problems arising in natural language processing can be modeled as a sequence labeling problem. Gaussian processes (GPs) provide a Bayesian approach to learning such problems in a kernel based framework. We develop Gaussian process models based on pseudo-likelihood to solve sequence labeling problems. The pseudo-likelihood model enables one to capture multiple dependencies among the output components of the sequence without becoming computationally intractable. We use an efficient variational Gaussian approximation method to perform inference in the proposed model. We also provide an iterative algorithm which can effectively make use of the information from the neighboring labels to perform prediction. The ability to capture multiple dependencies makes the proposed approach useful for a wide range of sequence labeling problems. Numerical experiments on some sequence labeling problems in natural language processing demonstrate the usefulness of the proposed approach.

Keywords: Gaussian processes, sequence labeling, variational inference

1 Introduction

Sequence labeling is the task of classifying a sequence of inputs into a sequence of outputs. It arises commonly in natural language processing (NLP) tasks such as part-of-speech tagging, chunking, named entity recognition etc. For instance, in part-of-speech (POS) tagging, the input is a sentence and the output is a sequence of POS tags. The output consists of components whose labels depend on the labels of other components in the output. Sequence labeling takes into account these inter-dependencies among various components of the output [17].

In recent years, sequence labeling has received considerable attention from the machine learning community and is often studied under the general framework of structured prediction. Many algorithms have been proposed to tackle sequence labeling problems. Hidden Markov model (HMM) [20], conditional random field (CRF) [13] and structural support vector machine (SSVM) [25] are the popular algorithms for sequence

labeling. SSVM allows learning a SVM for predicting a structured output including sequences. It is based on a large margin framework and is not probabilistic in nature. HMM is a probabilistic directed graphical model based on Markov assumption and has been widely used for problems in speech and language processing. CRF is also a probabilistic model based on Markov random field assumption. These parametric approaches can provide an estimate of uncertainty in predictions due to their probabilistic nature. However, they do not follow a Bayesian approach as they make a pointwise estimate of their parameters. This makes them less robust and heavily dependent on cross-validation for model selection. Bayesian CRF [19] overcomes this problem by providing a Bayesian treatment to CRF. Approaches like SSVM and maximum margin Markov network (M3N) make use of kernel functions which overcome the limitations arising due to the parametric nature of models such as CRF. Kernel CRF [14] is proposed to overcome this limitation of the CRF, but it is also not a Bayesian approach.

Gaussian processes (GPs) [21] have emerged as a better alternative to offer a non-parametric fully Bayesian approach to solve the sequence labeling problem. An initial work which studied Gaussian process for sequence labeling is [1], where GPs were proposed as an alternative to overcome the limitations of CRF; however they used a maximum a posteriori (MAP) approach instead of a fully Bayesian approach. This caused problems of model selection and robustness issues. A more recent work GPstruct [7] provides a Bayesian approach to general structured prediction problem with GPs. It uses Markov Chain Monte Carlo (MCMC) method to obtain the posterior distribution which slows down the inference. Their approach is based on Markov random field assumption which could not capture long range dependencies among the labels. This difficulty is overcome in [8] which uses an approximate likelihood to reduce the computational complexity arising due to the consideration of larger dependencies. In [8], the proposed model was used to solve grid structured problems in computer vision and was found to be effective in these problems.

In this work, we develop a Gaussian process approach based on pseudo-likelihood to solve sequence labeling problems (which we call GPSL). The GPSL model helps to capture multiple dependencies among the output components in a sequence without becoming computationally intractable. We develop a variational inference method to obtain the posterior which is faster than MCMC based approaches and does not suffer from convergence problems. We also provide an efficient algorithm to perform prediction in the GPSL model which effectively takes into account the dependence on multiple output components. We consider various GPSL models which consider different number of dependencies. We study the usefulness of these models on various sequence labeling problems arising in natural language processing (NLP). The GPSL models which capture more dependencies are found to be useful for these sequence labeling problems. They are also useful in sequence labeling data sets where the labels might be missing for some output components, for example, when the labels are obtained using crowd-sourcing. The main contributions of the paper are as follows :

1. A faster training algorithm based on variational inference.
2. An efficient prediction algorithm which considers multiple dependencies.
3. Application to sequence labeling problems in NLP.

The rest of the paper is organized as follows. Gaussian processes are introduced in Section 2. Section 3 discusses the proposed approach, Gaussian process sequence labeling (GPSL), in detail. We provide details of the variational inference and prediction algorithm for the GPSL model in Section 4 and Section 5 respectively. In Section 6, we study the performance of various GPSL models on sequence labeling problems and draw several conclusions in Section 7.

Notations: We consider a sequence labeling problem over sequences of input-output space pair $(\mathcal{X}, \mathcal{Y})$. The input sequence space \mathcal{X} is assumed to be made up of L components $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \dots \times \mathcal{X}_L$ and the associated output sequence space has L components $\mathcal{Y} = \mathcal{Y}_1 \times \mathcal{Y}_2 \times \dots \times \mathcal{Y}_L$. We assume a one-to-one mapping between the input and output components. Each component of the output space is assumed to take a discrete value from the set $\{1, 2, \dots, J\}$. Each component in the input space is assumed to belong to a P dimensional space \mathcal{R}^P representing features for that input component. Consider a collection of N training input-output examples $\mathbf{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$, where each example $(\mathbf{x}_n, \mathbf{y}_n)$ is such that $\mathbf{x}_n \in \mathcal{X}$ and $\mathbf{y}_n \in \mathcal{Y}$. Thus, \mathbf{x}_n consists of L components $(\mathbf{x}_{n1}, \mathbf{x}_{n2}, \dots, \mathbf{x}_{nL})$ and \mathbf{y}_n consists of L components $(y_{n1}, y_{n2}, \dots, y_{nL})$. The training data \mathbf{D} contains NL input-output components.

2 Background

A Gaussian process (GP) is a collection of random variables with the property that the joint distribution of any finite subset of which is a Gaussian [21]. It generalizes Gaussian distribution to infinitely many random variables and is used as a prior over a latent function. The GP is completely specified by a mean function and a covariance function. The covariance function is defined over latent function values of a pair of inputs and is evaluated using the Mercer kernel function over the pair of inputs. The covariance function expresses some general properties of functions such as their smoothness, and length-scale. A commonly used covariance function is the squared exponential (SE) or the Gaussian kernel

$$\text{cov}(f(\mathbf{x}_{mi}), f(\mathbf{x}_{nl})) = K(\mathbf{x}_{mi}, \mathbf{x}_{nl}) = \sigma_f^2 \exp(-\frac{\kappa}{2} \|\mathbf{x}_{mi} - \mathbf{x}_{nl}\|^2). \quad (1)$$

Here $f(\mathbf{x}_{mi})$ and $f(\mathbf{x}_{nl})$ are latent function values associated with the input components \mathbf{x}_{mi} and \mathbf{x}_{nl} respectively. $\boldsymbol{\theta} = (\sigma_f^2, \kappa)$ denotes the hyper parameters associated with the covariance function K .

Multi-class classification approaches are useful when the output consists of a single component taking values from a finite discrete set $\{1, 2, \dots, J\}$. Gaussian process multi-class classification approaches [26,10,9] associate a latent function f^j with every label $j \in \{1, 2, \dots, J\}$. Let the vector of latent function values associated with a particular label j over all the training examples be \mathbf{f}^j . The latent function f^j is assigned an independent GP prior with zero mean and covariance function K^j with hyper parameters $\boldsymbol{\theta}_j$. Thus, $\mathbf{f}^j \sim N(0, \mathbf{K}^j)$, where \mathbf{K}^j is a matrix obtained by evaluating the covariance function K^j over all the pairs of training data input components.

In multi-class classification, the likelihood over a multi-class output y_{nl} for an input \mathbf{x}_{nl} given the latent functions is defined as [21]

$$p(y_{nl}|f^1(\mathbf{x}_{nl}), f^2(\mathbf{x}_{nl}), \dots, f^J(\mathbf{x}_{nl})) = \frac{\exp(f^{y_{nl}}(\mathbf{x}_{nl}))}{\sum_{j=1}^J \exp(f^j(\mathbf{x}_{nl}))}. \quad (2)$$

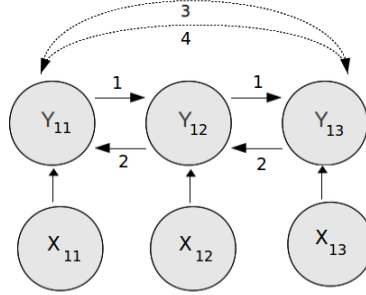
The likelihood (2) is known as multinomial logistic or softmax function and is used widely for the GP multi-class classification problems [26,9]. It is important to note that the likelihood function (2) used for the multi-class classification problems is not Gaussian. Hence, the posterior over the latent functions cannot be obtained in a closed form. GP multi-class classification approaches work by approximating the posterior as a Gaussian using approximate inference techniques such as Laplace approximation [26] and variational inference [10,9]. The Gaussian approximated posterior is then used to make predictions on the test data points. These approximations also yield an approximate marginal likelihood or a lower bound on marginal likelihood which can be used to perform model selection [21].

A sequence labeling problem can be treated as a multi-class classification problem. One can use multi-class classification to obtain a label for each component of the output independently. But this fails to take into account the inter-dependence among components. If one considers the entire output as a distinct class, then there would be an exponential number of classes and the learning problem becomes intractable. Hence, the sequence labeling problem has to be studied separately from the multi-class classification problems.

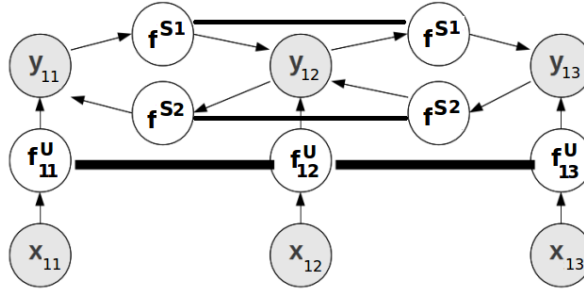
3 Gaussian Process Sequence Labeling

Most of the previous approaches [13,7] to sequence labeling use likelihood based on Markov random field assumption which captures only the interaction between neighboring output components. Non-neighboring components also play a significant role in problems such as sequence labeling. In these models, capturing such interactions are computationally expensive due to large clique size. The proposed approach, Gaussian process sequence labeling (GPSL), can take into account interactions among various output components without becoming computationally intractable by using a pseudo-likelihood (PL) model [4].

The PL model defines the likelihood of an output \mathbf{y}_n given the input \mathbf{x}_n as $p(\mathbf{y}_n|\mathbf{x}_n) \propto \prod_{l=1}^L p(y_{nl}|\mathbf{x}_{nl}, \mathbf{y}_n \setminus y_{nl})$. where, $\mathbf{y}_n \setminus y_{nl}$ represents all labels in \mathbf{y}_n except y_{nl} . PL models have been successfully used to address many sequence labeling problems in natural language processing [24,23]. They can capture long range dependencies without becoming computationally intractable as the normalization is done for each output component separately. In models such as CRF, normalization is done over the entire output. This renders them incapable of capturing long range dependencies as the number of summations in the normalization grows exponentially. The PL model is different from a locally normalized model like maximum entropy Markov model (MEMM) as each output component depends on several other output components. Therefore, they



(a) Dependence among input and output components. Dependence on various output components are modelled separately.



(b) Dependence of local and dependent latent functions. The local latent functions are defined over input-output pairs and dependent latent functions are defined between output components.

Fig. 1. Dependence of latent functions and input-output components in Gaussian process sequence labeling model.

do not suffer from the label bias problem [17] unlike MEMM. However, PL models create cyclic dependencies among the output components [11] and this makes prediction hard. We discuss an efficient approach to perform prediction in this case in Section 5.

The label of an output component need not depend on the labels of all the other output components. The dependencies among these output components are captured through the set S . Consider the directed graph in Figure 1a for a sequence labeling problem, where each output component is assumed to depend only on the neighboring output components. Here, the dependency set $S = \{1, 2\}$, where 1 denotes the dependence of an output component on the previous output component and 2 denotes its dependence on the next output component. One can also consider a model where an output component depends on the previous two output components and the next two output components. Let R denote the number of dependency relations in a set S (that is, R is the cardinality of S) and we assume it to be the same for all the output components for the sake of clarity in presentation. Taking into account those dependencies, we can

redefine the likelihood as

$$p(\mathbf{y}_n | \mathbf{x}_n) \propto \prod_{l=1}^L p(y_{nl} | \mathbf{x}_{nl}, \mathbf{y}_{nl}^S). \quad (3)$$

Here, \mathbf{y}_{nl}^S denotes the set of labels $\{y_{nl}^d\}_{d=1}^R$ of the output components referred by the dependency set S and y_{nl}^d denotes the label of the d^{th} dependent output component. In (3), instead of conditioning on the rest of the labels, we condition y_{nl} only on the labels defined by the dependency set S .

Now, the likelihood $p(y_{nl} | \mathbf{x}_{nl}, \mathbf{y}_{nl}^S)$ can be defined using a set of latent functions. We use different latent functions to model different dependencies. The dependency of the label y_{nl} on \mathbf{x}_{nl} is defined as a local dependency and is modeled as in GP multi-class classification. We associate a latent function with each label in the set $\{1, 2, \dots, J\}$. The latent function associated with a label j , denoted as f^{Uj} , is called a local latent function. It is defined over all the training input components \mathbf{x}_{nl} for every n and l and the latent function values associated with a particular label j over NL training examples are denoted by \mathbf{f}^{Uj} . The local latent functions associated with a particular input component \mathbf{x}_{nl} are denoted as $\mathbf{f}_{nl}^U = \{f_{nl}^{U1}, \dots, f_{nl}^{UJ}\}$. We also associate a latent function f^{Sd} with each dependency relation $d \in S$ and call them dependent latent functions. These latent functions are defined over all the values of a pair of labels (\hat{y}_{nl}, y_{nl}) where $\hat{y}_{nl} \in \{1, 2, \dots, J\}$ and $y_{nl} \in \{1, 2, \dots, J\}$. The latent function values associated with a particular dependency d over J^2 label pair values are denoted by \mathbf{f}^{Sd} . The dependence of various latent functions on the input and output components for the directed graph in Figure 1a is depicted in Figure 1b. Given these latent functions we define the likelihood $p(y_{nl} | \mathbf{x}_{nl}, \mathbf{y}_{nl}^S)$ to be a member of an exponential family:

$$p(y_{nl} | \mathbf{x}_{nl}, \mathbf{y}_{nl}^S, \{\mathbf{f}^{Uj}\}_{j=1}^J, \{\mathbf{f}^{Sd}\}_{d=1}^R) = \frac{\exp(f^{Uy_{nl}}(\mathbf{x}_{nl}) + \sum_{d=1}^R f^{Sd}(y_{nl}^d, y_{nl}))}{\sum_{y_{nl}=1}^J \exp(f^{Uy_{nl}}(\mathbf{x}_{nl}) + \sum_{d=1}^R f^{Sd}(y_{nl}^d, y_{nl}))}. \quad (4)$$

This differs from the softmax likelihood (2) used in multi-class classification in that it captures the dependencies among output components. Given the latent functions and the input $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$, the likelihood of the output $\mathbf{Y} = \{\mathbf{y}_n\}_{n=1}^N$ is

$$p(\mathbf{Y} | \mathbf{X}, \{\mathbf{f}^{Uj}\}_{j=1}^J, \{\mathbf{f}^{Sd}\}_{d=1}^R) = \prod_{n=1}^N \prod_{l=1}^L p(y_{nl} | \mathbf{x}_{nl}, \mathbf{y}_{n\{D_{nl}\}}, \{\mathbf{f}^{Uj}\}_{j=1}^J, \{\mathbf{f}^{Sd}\}_{d=1}^R) \quad (5)$$

We impose independent GP priors over the latent functions $\{f^{Uj}\}_{j=1}^J, \{f^{Sd}\}_{d=1}^R$. The latent function f^{Uj} is given a zero mean GP prior with covariance function K^{Uj} parameterized by θ_j . Thus, \mathbf{f}^{Uj} is a Gaussian with mean 0 and covariance \mathbf{K}^{Uj} of size $NL \times NL$, that is $p(\mathbf{f}^{Uj}) = \mathcal{N}(\mathbf{f}^{Uj}; \mathbf{0}, \mathbf{K}^{Uj})$. \mathbf{K}^{Uj} consists of covariance function evaluations over all the pairs of training data input components $\{\{\mathbf{x}_{nl}\}_{l=1}^L\}_{n=1}^N$. The latent function f^{Sd} is given zero mean GP prior with an identity covariance which is defined to be 1 when inputs are the same and 0 otherwise. Thus \mathbf{f}^{Sd} is a Gaussian with mean 0 and covariance \mathbf{I} of size J^2 , that is $p(\mathbf{f}^{Sd}) = \mathcal{N}(\mathbf{f}^{Sd}; \mathbf{0}, \mathbf{I}_{J^2})$. Let

$\mathbf{f}^{\mathbf{U}} = (\mathbf{f}^{\mathbf{U}1}, \mathbf{f}^{\mathbf{U}2}, \dots, \mathbf{f}^{\mathbf{U}J})$ be the collection of all local latent functions and $\mathbf{f}^{\mathbf{S}} = (\mathbf{f}^{\mathbf{S}1}, \mathbf{f}^{\mathbf{S}2}, \dots, \mathbf{f}^{\mathbf{S}R})$ be the collection of all dependent latent functions. Then the prior over $\mathbf{f}^{\mathbf{U}}$ and $\mathbf{f}^{\mathbf{S}}$ is defined as

$$p(\mathbf{f}^{\mathbf{U}}, \mathbf{f}^{\mathbf{S}} | \mathbf{X}) = \mathcal{N} \left(\begin{bmatrix} \mathbf{f}^{\mathbf{U}} \\ \mathbf{f}^{\mathbf{S}} \end{bmatrix} ; \mathbf{0}, \begin{bmatrix} \mathbf{K}^{\mathbf{U}} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}^{\mathbf{S}} \end{bmatrix} \right), \quad (6)$$

where $\mathbf{K}^{\mathbf{U}} = \text{diag}(\mathbf{K}^{\mathbf{U}1}, \mathbf{K}^{\mathbf{U}2}, \dots, \mathbf{K}^{\mathbf{U}J})$ is a block diagonal matrix and $\mathbf{K}^{\mathbf{S}} = \mathbf{I}_{J^2} \otimes \mathbf{I}_R$.

The posterior over the latent functions $p(\mathbf{f}^{\mathbf{U}}, \mathbf{f}^{\mathbf{S}} | \mathbf{D})$ is

$$p(\mathbf{f}^{\mathbf{U}}, \mathbf{f}^{\mathbf{S}} | \mathbf{X}, \mathbf{Y}) = \frac{1}{p(\mathbf{Y} | \mathbf{X})} p(\mathbf{Y} | \mathbf{X}, \mathbf{f}^{\mathbf{U}}, \mathbf{f}^{\mathbf{S}}) p(\mathbf{f}^{\mathbf{U}}, \mathbf{f}^{\mathbf{S}} | \mathbf{X})$$

where $p(\mathbf{Y} | \mathbf{X}) = \int p(\mathbf{Y} | \mathbf{X}, \mathbf{f}^{\mathbf{U}}, \mathbf{f}^{\mathbf{S}}) p(\mathbf{f}^{\mathbf{U}}, \mathbf{f}^{\mathbf{S}} | \mathbf{X}) d\mathbf{f}^{\mathbf{U}} d\mathbf{f}^{\mathbf{S}}$ is called evidence. Evidence is a function of hyper-parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \dots, \boldsymbol{\theta}_J)$ and is maximized to estimate them. For notational simplicity, we suppress the dependence of evidence, posterior and prior on the hyper-parameter $\boldsymbol{\theta}$. Due to the non-Gaussian nature of the likelihood, evidence is intractable and the posterior cannot be determined exactly. We use a variational inference technique to obtain an approximate posterior. Variational inference is faster than sampling based techniques used in [7] and does not suffer from convergence problems [16]. It can easily handle multi-class problems and is scalable to models with a large number of parameters. Further, it provides an approximation to the evidence which is useful in estimating the hyper-parameters of the model.

4 Variational Inference

A variational Inference technique [16] approximates the intractable posterior by an approximate variational distribution. It approximates the posterior $p(\mathbf{f} | \mathbf{X}, \mathbf{Y})$ by a variational distribution $q(\mathbf{f} | \boldsymbol{\gamma})$, where $\mathbf{f} = (\mathbf{f}^{\mathbf{U}}, \mathbf{f}^{\mathbf{S}})$ and $\boldsymbol{\gamma}$ represents the variational parameters. In variational inference, this is done by minimizing the Kullback-Leibler (KL) divergence between $q(\mathbf{f} | \boldsymbol{\gamma})$ and $p(\mathbf{f} | \mathbf{X}, \mathbf{Y})$. This is often intractable and the variational parameters are obtained by maximizing a variational lower bound $L(\boldsymbol{\theta}, \boldsymbol{\gamma})$.

$$KL(q(\mathbf{f} | \boldsymbol{\gamma}) || p(\mathbf{f} | \mathbf{X}, \mathbf{Y})) = -L(\boldsymbol{\theta}, \boldsymbol{\gamma}) + \log p(\mathbf{Y} | \mathbf{X}) \quad (7)$$

$$\text{where } L(\boldsymbol{\theta}, \boldsymbol{\gamma}) = -KL(q(\mathbf{f} | \boldsymbol{\gamma}) || p(\mathbf{f} | \mathbf{X})) + \int q(\mathbf{f} | \boldsymbol{\gamma}) \log p(\mathbf{Y} | \mathbf{X}, \mathbf{f}) d\mathbf{f}.$$

Maximizing the variational lower bound $L(\boldsymbol{\theta}, \boldsymbol{\gamma})$ results in minimizing the KL divergence $KL(q(\mathbf{f} | \boldsymbol{\gamma}) || p(\mathbf{f} | \mathbf{X}, \mathbf{Y}))$, since the evidence $p(\mathbf{Y} | \mathbf{X})$ does not depend on the variational parameters.

We use a variational Gaussian (VG) approximate inference approach [18] where the variational distribution is assumed to be a Gaussian. Variational Gaussian approaches can be slow because of the requirement to estimate the covariance matrix. Fortunately, recent advances in VG inference approaches [18] enable one to compute the covariance matrix using $\mathcal{O}(NL)$ variational parameters. In fact, we use the VG approach for GPs [12] which requires computation of only $\mathcal{O}(NL)$ variational parameters, but at the

same time uses a concave variational lower bound. We assume the variational distribution $q(\mathbf{f}|\gamma)$ takes the form of a Gaussian distribution and factorizes as $q(\mathbf{f}^U|\gamma^U)q(\mathbf{f}^S|\gamma^S)$ where $\gamma = \{\gamma^U, \gamma^S\}$. Let $q(\mathbf{f}^U|\gamma^U) = \mathcal{N}(\mathbf{f}^U; \mathbf{m}^U, \mathbf{V}^U)$ where $\gamma^U = \{\mathbf{m}^U, \mathbf{V}^U\}$ and $q(\mathbf{f}^S) = \mathcal{N}(\mathbf{f}^S; \mathbf{m}^S, \mathbf{V}^S)$ where $\gamma^S = \{\mathbf{m}^S, \mathbf{V}^S\}$. Then, the variational lower bound $L(\theta, \gamma)$ can be written as

$$L(\theta, \gamma) = \frac{1}{2}(\log |\mathbf{V}^U \Omega^U| + \log |\mathbf{V}^S \Omega^S| - \text{tr}(\mathbf{V}^U \Omega^U) - \text{tr}(\mathbf{V}^S \Omega^S) - \mathbf{m}^{U\top} \Omega^U \mathbf{m}^U - \mathbf{m}^{S\top} \Omega^S \mathbf{m}^S) + \sum_{n=1}^N \sum_{l=1}^L \mathbb{E}_{q(\mathbf{f}^U|\gamma^U)q(\mathbf{f}^S|\gamma^S)}[\log p(y_{nl}|\mathbf{x}_{nl}, \mathbf{y}_{nl}^S, \mathbf{f})] \quad (8)$$

where $\Omega^U = \mathbf{K}^{U-1}$, $\Omega^S = \mathbf{K}^{S-1}$ and $\mathbb{E}_{q(x)}[f(x)] = \int f(x)q(x)dx$ represents the expectation of $f(x)$ with respect to the density $q(x)$. Since \mathbf{K}^U is block diagonal, its inverse is block diagonal, and hence Ω^U is block diagonal that is $\Omega^U = \text{diag}(\Omega^{U1}, \Omega^{U2}, \dots, \Omega^{UJ})$, where $\Omega^{Uj} = \mathbf{K}^{Uj-1}$. Similarly, Ω^S is also a block diagonal with each block being a diagonal matrix \mathbf{I}_{J^2} . The marginal variational distribution of local latent function values \mathbf{f}^{Uj} is a Gaussian with mean \mathbf{m}^{Uj} and covariance \mathbf{V}^{Uj} , and that of dependent latent function values \mathbf{f}^{Sd} is a Gaussian with mean \mathbf{m}^{Sd} and covariance \mathbf{V}^{Sd} . The variational lower bound $L(\theta, \gamma)$ requires computing an expectation of the log likelihood with respect to the variational distribution. However, the integral is intractable since the likelihood is a softmax function. So, we use Jensen's inequality to obtain a tractable lower bound to the expectation of log likelihood. The variational lower bound $L(\theta, \gamma)$ can be written as

$$\begin{aligned} & \frac{1}{2} \left(\sum_{j=1}^J (\log |\mathbf{V}^{Uj} \Omega^{Uj}| - \text{tr}(\mathbf{V}^{Uj} \Omega^{Uj}) - \mathbf{m}^{Uj\top} \Omega^{Uj} \mathbf{m}^{Uj}) \right. \\ & \quad \left. + \sum_{d=1}^R (\log |\mathbf{V}^{Sd} \Omega^{Sd}| - \text{tr}(\mathbf{V}^{Sd} \Omega^{Sd}) - \mathbf{m}^{Sd\top} \Omega^{Sd} \mathbf{m}^{Sd}) \right) \\ & + \sum_{n=1}^N \sum_{l=1}^L \left(m_{nl}^{Uy_{nl}} + \sum_{d=1}^R m_{(y_{nl}^d, y_{nl})}^{Sd} - \log \left(\sum_{q=1}^J \exp(m_{nl}^{Uj} + \frac{1}{2} V_{(nl, nl)}^{Uj} \right. \right. \\ & \quad \left. \left. + \sum_{d=1}^R m_{(y_{nl}^d, q)}^{Sd} + \frac{1}{2} V_{((y_{nl}^d, q), (y_{nl}^d, q))}^{Sd}) \right) \right). \quad (9) \end{aligned}$$

The variational parameters $\gamma = \{\{\mathbf{m}^{Uj}\}_{j=1}^J, \{\mathbf{V}^{Uj}\}_{j=1}^J, \{\mathbf{m}^{Sd}\}_{d=1}^R, \{\mathbf{V}^{Sd}\}_{d=1}^R\}$ are estimated by maximizing the variational lower bound (9). The lower bound is jointly concave with respect to all the variational parameters [6] and the optimum can be easily found using gradient based optimization techniques.

The variational parameters are estimated using a co-ordinate ascent approach. We repeatedly estimate each variational parameter while keeping the others fixed. The variational mean parameters \mathbf{m}^{Uj} and \mathbf{m}^{Sd} are estimated using gradient based approaches. The variational covariance matrices \mathbf{V}^{Uj} and \mathbf{V}^{Sd} are estimated under the positive semi-definite (p.s.d.) constraint. This can be done efficiently using the fixed point approach mentioned in [12]. It is reported to converge faster than other VG approaches

Algorithm 1 Model selection and learning in Gaussian process sequence labeling model

```

1: Input: Training data  $(\mathbf{X}, \mathbf{Y})$ , dependency set  $S$ 
2: Initialize hyper-parameters  $\theta$ , variational parameters  $\gamma$ 
3: repeat
4:   repeat
5:     for  $j = 1$  to  $J$  do
6:       Update  $\mathbf{m}^{\text{Uj}}$  by maximizing (9) w.r.t  $\mathbf{m}^{\text{Uj}}$ 
7:       Update  $\mathbf{V}^{\text{Uj}}$  by maximizing (9) w.r.t  $\mathbf{V}^{\text{Uj}}$ 
8:     end for
9:     for  $d = 1$  to  $R$  do
10:      Update  $\mathbf{m}^{\text{Sd}}$  by maximizing (9) w.r.t  $\mathbf{m}^{\text{Sd}}$ 
11:      Update  $\mathbf{V}^{\text{Sd}}$  by maximizing (9) w.r.t  $\mathbf{V}^{\text{Sd}}$ 
12:    end for
13:   until relative increase in lower bound (9) is small
14:   Update  $\theta$  by maximizing (9) w.r.t  $\theta$ 
15: until relative increase in lower bound (9) is small
16: Return:  $\theta, \gamma$ 

```

for GPs and is based on a concave objective function similar to (9). The approach maintains the p.s.d. constraint on the covariance matrix and computes \mathbf{V}^{Uj} by estimating only $\mathcal{O}(NL)$ variational parameters. Estimation of \mathbf{V}^{Uj} using the fixed point approach converges since (9) is strictly concave with respect to \mathbf{V}^{Uj} . The variational covariance matrix \mathbf{V}^{Sd} is diagonal since Ω^{Sd} is diagonal. Hence, for computing a p.s.d. \mathbf{V}^{Sd} we need to estimate only the diagonal elements of \mathbf{V}^{Sd} under the element-wise non-negativity constraint. This can be done easily using gradient based methods. The variational parameters γ are estimated for a particular set of hyper-parameters θ . The hyper-parameters θ are also estimated by maximizing the lower bound (9). The variational parameters γ and the model parameters θ are estimated alternately following a variational expectation maximization (EM) approach [16]. Algorithm 1 summarizes various steps involved in our approach.

The variational lower bound (9) is strictly concave with respect to each of the variational parameters. Hence, the estimation of variational parameters using co-ordinate ascent algorithm (inner loop) converges [3]. Convergence of EM for exponential family guarantees the convergence of Algorithm 1. The overall computational complexity of Algorithm 1 is dominated by the computation of \mathbf{V}^{Uj} . It takes $\mathcal{O}(JN^3L^3)$ time as it requires inversion of J covariance matrices of size $NL \times NL$. The computational complexity for estimating \mathbf{V}^{Sd} is $\mathcal{O}(RNLJ)$ and is negligible compared to the estimation of \mathbf{V}^{Uj} . Note that the computational complexity of the algorithm increases linearly with respect to the number of dependencies R .

5 Prediction

We propose an iterative prediction algorithm which can effectively take into account the presence of multiple dependencies. The variational posterior distributions estimated

using VG approximation $q(\mathbf{f}^U) = \prod_{j=1}^J q(\mathbf{f}^{Uj}) = \prod_{j=1}^J \mathcal{N}(\mathbf{f}^{Uj}; \mathbf{m}^{Uj}, \mathbf{V}^{Uj})$ and $q(\mathbf{f}^S) = \prod_{d=1}^R q(\mathbf{f}^{Sd}) = \prod_{d=1}^R \mathcal{N}(\mathbf{f}^{Sd}; \mathbf{m}^{Sd}, \mathbf{V}^{Sd})$ can be used to predict a test output sequence \mathbf{y}_* given a test input sequence \mathbf{x}_* . The predictive probability of assigning a label y_{*l} to a component of the output \mathbf{y}_* , given \mathbf{x}_{*l} and rest of the labels $\mathbf{y}_* \setminus y_{*l}$ is

$$\begin{aligned} p(y_{*l} | \mathbf{x}_{*l}, \mathbf{y}_* \setminus y_{*l}) &= \int p(y_{*l} | \mathbf{x}_{*l}, \mathbf{y}_* \setminus y_{*l}, \mathbf{f}_*) p(\mathbf{f}_*) d\mathbf{f}_* \\ &= \int \frac{\exp(f_{*l}^{Uy_{*l}} + \sum_{d=1}^R f_*^{Sd}(y_{*l}^d, y_{*l}))}{\sum_{y_{*l}=1}^J \exp(f_{*l}^{Uy_{*l}} + \sum_{d=1}^R f_*^{Sd}(y_{*l}^d, y_{*l}))} \\ &\quad \{p(f_{*l}^{Uj})\}_{j=1}^J \{p(f_*^{Sd})\}_{d=1}^R \{df_{*l}^{Uj}\}_{j=1}^J \{df_*^{Sd}\}_{d=1}^R \quad (10) \end{aligned}$$

where $p(\mathbf{f}_*)$ denotes the predictive distribution of all the latent function values for the test input \mathbf{x}_* . In (10), $p(f_{*l}^{Uj})$ represents the predictive distribution of the local latent function j for a test input component \mathbf{x}_{*l} . This is Gaussian with mean m_{*l}^{Uj} and variance v_{*l}^{Uj} where,

$$\begin{aligned} m_{*l}^{Uj} &= \mathbf{K}_{*l}^{Uj \top} \boldsymbol{\Omega}^{Uj} \mathbf{m}^{Uj} \quad \text{and} \\ v_{*l}^{Uj} &= K_{*l,*l}^{Uj} - \mathbf{K}_{*l}^{Uj \top} (\boldsymbol{\Omega}^{Uj} - \boldsymbol{\Omega}^{Uj} \mathbf{V}^{Uj} \boldsymbol{\Omega}^{Uj}) \mathbf{K}_{*l}^{Uj}. \end{aligned}$$

Here, \mathbf{K}_{*l}^{Uj} is an NL dimensional vector obtained from the kernel evaluations for the label j between the test input data component \mathbf{x}_{*l} and the training data \mathbf{X} and $K_{*l,*l}^{Uj}$ represents the kernel evaluation of the test data input component \mathbf{x}_{*l} with itself. \mathbf{f}^{Sd} is independent of the test data input and the predictive distribution $p(\mathbf{f}^{Sd})$ is the same as $p(\mathbf{f}^{Sd})$. This is a Gaussian with mean \mathbf{m}^{Sd} and covariance \mathbf{V}^{Sd} . The computation of the expected value of softmax with respect to the latent functions (10) is intractable. Instead we compute softmax of the expected value of the latent functions and compute a normalized probabilistic score. We refine the normalized score to take into account the uncertainty in true labels associated with the dependencies and compute the refined normalized score (RNS) as

$$RNS(y_{*l}, \mathbf{x}_{*l}) = \frac{\exp(m_{*l}^{Uy_{*l}} + \frac{1}{2}v_{*l}^{Uy_{*l}} + \sum_{d=1}^R \mathbb{E}_{y_{*l}^d} [g^d(y_{*l}^d, y_{*l})])}{\sum_{q=1}^J \exp(m_{*l}^{Uq} + \frac{1}{2}v_{*l}^{Uq} + \sum_{d=1}^R \mathbb{E}_{y_{*l}^d} [g^d(y_{*l}^d, q)])}$$

Here, $g^d(y^d, y) = \mathbf{m}_{(y^d, y)}^{Sd} + \frac{1}{2} \mathbf{V}_{((y^d, y), (y^d, y))}^{Sd}$ determines the contribution of the label y^d of dependency d in predicting the output label y . RNS considers an expected value over all the possible labelings associated with a dependency d . The expectation is computed using the RNS value associated with the labels y_{*l}^d for the input x_{*l}^d , that is, $\mathbb{E}_{y_{*l}^d} [\cdot] = \sum_{y_{*l}^d=1}^J RNS(y_{*l}^d, x_{*l}^d) [\cdot]$.

We provide an iterative approach to estimate the labels of a test output in Algorithm 2. An initial RNS value is computed without considering the dependencies. We iteratively refine the RNS value using the previously computed RNS value by taking into account the dependencies. The process is continued until convergence. The final RNS value is used to make prediction separately for each output component by

Algorithm 2 Prediction in Gaussian process sequence labeling model

-
- 1: **Input:** Test data $\mathbf{x}_* = (\mathbf{x}_{*1}, \dots, \mathbf{x}_{*L})$, posterior mean $\{\mathbf{m}^{\text{Uj}}\}_{j=1}^J$ and $\{\mathbf{m}^{\text{Sd}}\}_{d=1}^R$ and posterior covariance $\{\mathbf{V}^{\text{Uj}}\}_{j=1}^J$ and $\{\mathbf{V}^{\text{Sd}}\}_{d=1}^R$
 - 2: Obtain predictive means $\{\{m_{*l}^{\text{Uj}}\}_{j=1}^J\}_{l=1}^L$, and variances $\{\{v_{*l}^{\text{Uj}}\}_{j=1}^J\}_{l=1}^L$
 - 3: **Initialize :** $RNS^0(y_{*l}, \mathbf{x}_{*l}) = \frac{\exp(m_{*l}^{\text{Uj}} + \frac{1}{2}v_{*l}^{\text{Uj}})}{\sum_{j=1}^J \exp(m_{*l}^{\text{Uj}} + \frac{1}{2}v_{*l}^{\text{Uj}})} \quad \forall y_{*l} = 1, \dots, J, \forall l = 1 \dots, L$
 - 4: **Initialize :** $t = 0$
 - 5: **repeat**
 - 6: $t = t + 1$
 - 7: **for** $l = 1$ **to** L **do**
 - 8: **for** $y_{*l} = 1$ **to** J **do**
 - 9: $RNS^t(y_{*l}, \mathbf{x}_{*l}) = \frac{\exp(m_{*l}^{\text{Uj}} + \frac{1}{2}v_{*l}^{\text{Uj}} + \sum_{d=1}^R \mathbb{E}_{y_{*l}^d} [g^d(y_{*l}^d, y_{*l})])}{\sum_{j=1}^J \exp(m_{*l}^{\text{Uj}} + \frac{1}{2}v_{*l}^{\text{Uj}} + \sum_{d=1}^R \mathbb{E}_{y_{*l}^d} [g^d(y_{*l}^d, q)])}$
 - 10: where $\mathbb{E}_{y_{*l}^d} [\cdot] = \sum_{y_{*l}^d=1}^J RNS^{t-1}(y_{*l}^d, \mathbf{x}_{*l}^d) [\cdot]$
 - 11: **end for**
 - 12: **end for**
 - 13: **until** change in RNS^t w.r.t RNS^{t-1} is small
 - 14: $(\hat{y}_{*1}, \dots, \hat{y}_{*L}) = (\arg\max_{y_{*1}} RNS^t(y_{*1}, \mathbf{x}_{*1}), \dots, \arg\max_{y_{*L}} RNS^t(y_{*L}, \mathbf{x}_{*L}))$
 - 15: **Return:** $(\hat{y}_{*1}, \dots, \hat{y}_{*L})$
-

assigning labels with the maximum RNS value. The computational complexity of Algorithm 2 is $\mathcal{O}(J^2RL)$ and is same as that of Viterbi algorithm [20] for a single dependency case. The convergence of Algorithm 2 follows from the analysis presented in [15] for a similar fixed point algorithm. The algorithm is found to converge in a few iterations in our experiments.

6 Experimental Results

We conduct experiments to study the generalization performance of the proposed Gaussian Process Sequence labeling (GPSL) model. We use the sequence labeling problems in natural language processing to study the behavior of the proposed approach. Although the proposed approach is general and can handle dependencies of any length, we consider three different models of the proposed approach in our experiments. The first model, GPSL1, assumes that the current label depends only on the previous label. The second model, GPSL2, assumes that the current label depends both on the previous and the next label in the sequence. The third model, GPSL4, assumes that the current label depends on the previous two labels and the next two labels.

We consider four sequence labeling problems in natural language processing to study the performance of the proposed approach. The datasets for all these problems are obtained from the CRF++⁴ toolbox. We provide a brief description of the tasks in each of these data sets.

Base NP : We need to identify noun phrases in a sentence. The starting word in the

⁴ Available at <http://crfpp.googlecode.com/svn/trunk/doc/index.html>

noun phrase is given a label B , while the words inside the noun phrase are given a label I . All the other words are given a label O . The task here is to assign each word with a label from the set $\{B, I, O\}$.

Chunking : Shallow parsing or chunking identifies constituents in a sentence such as noun phrase, verb phrase etc. Here, each word in a sentence is labeled as belonging to verb phrase, noun phrase etc. In the *Chunking* dataset, words are assigned a label from a set of size 14.

Segmentation : Segmentation is the process of finding meaningful segments in a text such as words, sentences etc. We consider a word segmentation problem where the words are identified from a Chinese sentence. The *Segmentation* data set assigns each unit in the sentence a label denoting whether it is beginning of a word (B) or inside a word (I). The task is to assign either of these two labels to each unit in a sentence.

Japanese NE : We need to perform Named Entity Recognition (NER) where the task is to identify whether the words in a sentence denote a named entity such as person, place, time etc. We use the *JapaneseNE* dataset where the Japanese words are assigned one of 17 different named entities.

In all these data sets except *Segmentation*, a sentence is considered as an input and words in the sentence as input components. In *Segmentation*, every alphabet is considered as an input component. The features for each input component are extracted using the template files provided in the CRF++ package. The properties of all the data sets are summarized in Table 1. It mentions the number of sentences (N) used for training and testing. The effective sample size (NL) for the GPSL models is obtained by multiplying this quantity by average sentence length which increases the data size by an order of magnitude.

We compare the performance of the proposed approach with popular sequence labeling approaches, structural SVM (SSVM) [2]⁵, conditional random field (CRF) [5]⁶, and GPstruct [7]⁷. All the models used a linear kernel. GPstruct experiments are run for 100000 elliptical slice sampling steps. The performance is measured in terms of average Hamming loss over all the test data points. The Hamming loss between the actual test output \mathbf{y}_* and the predicted test output $\hat{\mathbf{y}}_*$ is given by $Loss(\mathbf{y}_*, \hat{\mathbf{y}}_*) = \sum_{l=1}^L \mathbb{I}(y_{*l} \neq \hat{y}_{*l})$, where $\mathbb{I}(\cdot)$ is the indicator function. Table 1 compares the performance (percentage of the average Hamming loss) of various approaches on the four sequence labeling problems. The GPSL models, SSVM, CRF and GPstruct are run over 10 independent partitions of the data set⁸ and a mean of the Hamming loss over all the partitions along with the standard deviation are reported in Table 1.

The reported results show that the GPSL models with multiple dependencies performed better than GPstruct on *BaseNP* and *Segmentation*. In the other two data sets, GPSL models came close to GPstruct. We find that increasing the number of dependencies helped to improve the performance in general except for the *Segmentation* data set. This is due to the difference in nature of the sequence labeling task involved

⁵ Code available at http://drona.csa.iisc.ernet.in/~shirish/structsvm_sdm.html

⁶ Code available at http://leon.bottou.org/projects/sgd#stochastic_gradient_descent_version_2

⁷ Code available at <https://github.com/sebastien-bratieres/pygpstruct>

⁸ The train and test set partitions are different from those used by [7].

Table 1. Properties of the sequence labeling data sets and a comparison of the performance of various models on these data sets. The approaches GPSL1, GPSL2, GPSL4, SSVM, CRF and GPstruct are compared using average Hamming loss (in percentage). The numbers in bold face style indicate the best results among these approaches. ‘★’ and ‘†’ denote if the performance of a method is significantly different from the best performing method and GPstruct respectively, according to paired t-test with 5% significance level.

	Base NP	Chunking	Segmentation	Japanese NE
#labels	3	14	2	17
#features	6438	29764	1386	102,799
training/ test sentences	150/150	50/50	20/16	50/50
GPSL1	5.73±0.98★	13.02±1.87★	23.45±2.96	8.26±2.63★
GPSL2	5.55±0.92★	12.69±1.69★	23.51±2.93	7.86±2.45 ★
GPSL4	5.54±0.94★	12.70±1.79★	23.53±2.85	7.82±2.56 ★
CRF	5.21±0.84†	11.76±1.73★†	24.10±3.49★†	7.76±2.80 ★
SSVM	5.19±0.91†	10.71±1.49†	23.46±3.45	6.17±2.60†
GPstruct	5.66±0.93★	12.56±1.82★	23.55±2.90	7.79±2.92 ★

in segmentation. For other data sets, the GPSL model which considered both the previous and next label (GPSL2) gave a better performance. The performance of the GPSL model which considered the previous and the next 2 labels (GPSL4) improved only marginally or worsened compared to GPSL2 on these data sets. We note that increasing the number of dependencies beyond four did not bring any improvement in performance for the sequence labeling data sets that we have considered. Overall, the performance of the SSVM is found to be better than other approaches in these sequence labeling data sets. However, GPSL models have the advantage of being Bayesian and can provide a confidence over label predictions which is useful for many NLP tasks.

6.1 Runtime performance of the GPSL models

The proposed GPSL models are implemented in Matlab. The GPSL Matlab programs are run on a 3.2 GHz Intel processor with 4GB of shared main memory under Linux. The SSVM approach is implemented in C, the CRF approach is coded in C++ and the GPstruct approach is in Python. Since the implementation languages differ, it is unfair to make a runtime comparison of various approaches. Table 2 compares the average runtime (in seconds) for training various GPSL models and GPstruct on the sequence labeling data sets. We find that the GPSL models are an order of magnitude faster than GPstruct. We also find that increasing the dependencies resulted in only a slight increase in runtime.

6.2 Experiments with the Prediction algorithm

We conducted experiments to study the performance of Algorithm 2 used to make prediction. The algorithm is compared with the commonly used Viterbi algorithm [20] for the sequence labeling task. Viterbi algorithm consists of a forward phase which calculates the best value attained at the end of the sequence and a backward phase which

Table 2. Comparison of average running time (seconds) of various GPSL models and GPstruct

Data	GPSL1	GPSL2	GPSL4	GPstruct
Segmentation	17.13	19.64	22.83	3.82e+03
Chunking	1.09e+03	1.35e+03	1.71e+03	4.56e+04
Base NP	6.01e+03	6.69e+03	7.25e+03	7.54e+04
Japanese NE	1.24e+03	1.56e+03	1.93e+03	4.92e+04

Table 3. Comparison of the prediction algorithms using GPSL1 model

Data	average Hamming loss		paired t-test	average runtime (seconds)		average iterations
	Algorithm 2	Viterbi	t-value	Algorithm 2	Viterbi	Algorithm 2
Segmentation	23.45	24.26	3.8183	0.1227	0.0856	5
Chunking	13.02	13.69	3.6421	0.2491	0.2628	5
Base NP	5.73	5.75	0.3162	0.5207	0.5338	4
Japanese NE	8.26	8.84	2.475	0.3661	0.5653	3

finds the sequence of labels that lead to it. It is useful only for the setting where one considers a dependency with the previous label. Therefore, we study how the performance of the GPSL1 model differs when Viterbi algorithm is used for prediction instead of the proposed algorithm. We consider an implementation of the Viterbi algorithm provided by the UGM toolkit [22]. Table 3 compares the predictive and runtime performance of the two algorithms. We observe that Algorithm 2 gave a better predictive and runtime performance than the Viterbi algorithm. The predictive performance of Algorithm 2 is significantly better than Viterbi on *Segmentation*, *Chunking* and *JapaneseNE*. The t-values calculated using paired t-test on these data sets are found to be greater than the critical value of 2.262 for a level of significance 0.05 and 9 degrees of freedom. We also observed that Algorithm 2 converged in 3-5 iterations on an average.

6.3 Experiments with Missing Labels

In many sequence labeling tasks in NLP, the labels of some of the output components might be missing in the training data set. This is common when crowd sourcing techniques are employed to obtain the labels. Sequence labeling approaches such as SSVM and CRF are not readily applicable to data sets with missing labels. GPSL models are useful to learn from the data sets with missing labels due to their ability to capture larger dependencies. We learn the GPSL models from the sequence labeling data sets with some fraction of the labels missing. We vary the fraction of missing labels and study how the performance of our model varies with respect to missing labels. Figure 2 provides the variation in performance of various GPSL models as we vary the fraction of missing labels. The performance is measured in terms of accuracy which is obtained by subtracting the average Hamming loss from 1. We find that the performance of the GPSL models does not significantly degrade as the fraction of the missing labels increases. Figure 2 shows that GPSL4 which uses the previous and the next 2 labels provides a better performance than the other GPSL models. GPSL4 learns a better model by considering a larger neighborhood information and is useful to handle data sets with missing labels.

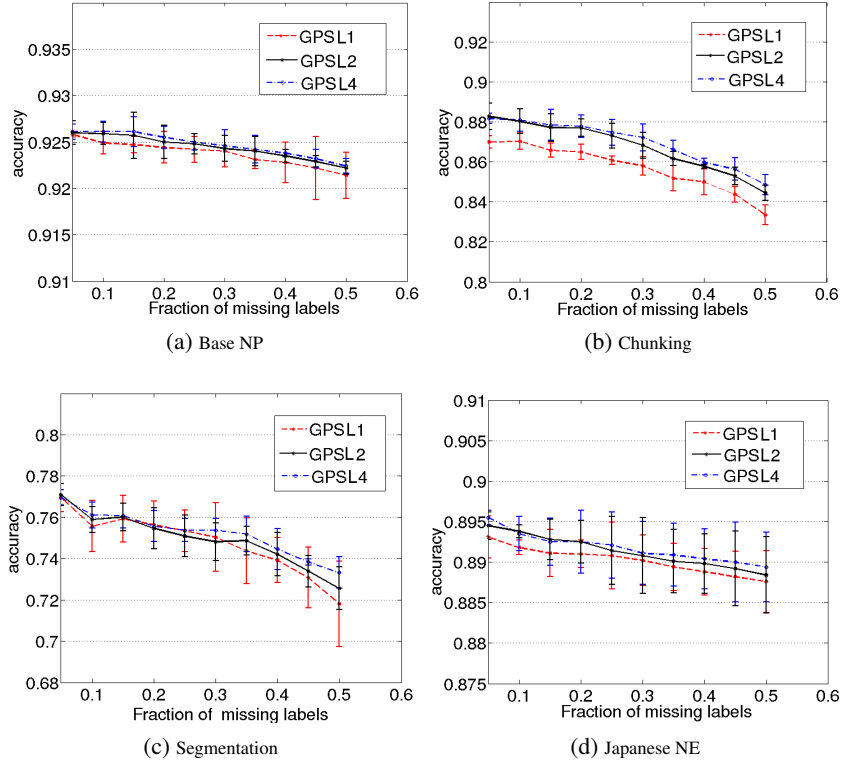


Fig. 2. Variation in accuracy as the fraction of missing labels is varied from 0.05 to 0.5

7 Conclusion

We proposed a novel Gaussian Process approach to perform sequence labeling based on pseudo-likelihood approximation. The use of pseudo-likelihood enabled the model to capture multiple dependencies without becoming computationally intractable. The approach used a faster inference scheme based on variational inference. We also proposed an approach to perform prediction which makes use of the information from the neighboring labels. The proposed approach is useful for a wide range of sequence labeling problems arising in natural language processing. Experimental results showed that GPSL models, which capture multiple dependencies, are useful in sequence labeling problems. The ability to capture multiple dependencies makes them effective in handling data sets with missing labels.

References

1. Altun, Y., Hofmann, T., Smola, A.J.: Gaussian Process Classification for Segmenting and Annotating Sequences. In: ICML (2004)

2. Balamurugan, P., Shevade, S., Sundararajan, S., Keerthi, S.: A Sequential Dual Method for Structural SVMs. In: SDM. pp. 223–234 (2011)
3. Bertsekas, D.P.: Nonlinear Programming. Athena Scientific (1999)
4. Besag, J.: Statistical analysis of non-lattice data. *The Statistician* 24, 179–195 (1975)
5. Bottou, L.: Large-Scale Machine Learning with Stochastic Gradient Descent. In: COMP-STAT (2010)
6. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press (2004)
7. Bratieres, S., Quadrianto, N., Ghahramani, Z.: Bayesian Structured Prediction Using Gaussian Processes. *IEEE transactions on Pattern Analysis and Machine Intelligence* (2014)
8. Bratieres, S., Quadrianto, N., Nowozin, S., Ghahramani, Z.: Scalable Gaussian Process Structured Prediction for Grid Factor Graph Applications. *ICML* (2014)
9. Chai, K.M.A.: Variational Multinomial Logit Gaussian Process. *J. Mach. Learn. Res.* 13 (2012)
10. Girolami, M., Rogers, S.: Variational Bayesian Multinomial Probit Regression with Gaussian Process Priors. *Neural Computation* 18(8), 1790–1817 (2006)
11. Heckerman, D., Chickering, D.M., Meek, C., Rounthwaite, R., Kadie, C.: Dependency Networks for Inference, Collaborative Filtering, and Data Visualization. *J. Mach. Learn. Res.* 1, 49–75 (2001)
12. Khan, M.E., Mohamed, S., Murphy, K.P.: Fast Bayesian Inference for Non-Conjugate Gaussian Process Regression. In: NIPS. pp. 3149–3157 (2012)
13. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: *ICML*. pp. 282–289 (2001)
14. Lafferty, J.D., Zhu, X., Liu, Y.: Kernel Conditional Random Fields: Representation and Clique Selection. In: *ICML* (2004)
15. Li, Q., Wang, J., Wipf, D.P., Tu, Z.: Fixed-Point Model For Structured Labeling. In: *ICML*. pp. 214–221 (2013)
16. Murphy, K.P.: Machine learning: A Probabilistic Perspective. The MIT Press (2012)
17. Noah, A.S.: Linguistic Structure Prediction. Morgan and Claypool (2011)
18. Opper, M., Archambeau, C.: The Variational Gaussian Approximation Revisited. *Neural Computation* 21, 786–792 (2009)
19. Qi, Y., Szummer, M., Minka, T.P.: Bayesian conditional random fields. *Proc. AISTATS* (2005)
20. Rabiner, L.R.: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE* 77(2), 257–286 (1989)
21. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). MIT Press (2005)
22. Schmidt, M.: UGM: A Matlab toolbox for probabilistic undirected graphical models. (2007), <http://www.cs.ubc.ca/~schmidtm/Software/UGM.html>
23. Sutton, C., McCallum, A.: Piecewise Pseudolikelihood for Efficient Training of Conditional Random Fields. pp. 863–870. *ICML* (2007)
24. Toutanova, K., Klein, D., Manning, C.D., Singer, Y.: Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In: *HLT-NAACL*. pp. 252–259 (2003)
25. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large Margin Methods for Structured and Interdependent Output Variables. *J. Mach. Learn. Res.* 6, 1453–1484 (2005)
26. Williams, C.K.I., Barber, D.: Bayesian Classification with Gaussian Processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(12), 1342–1351 (1998)